Question 1: Can you briefly explain the functionality of your shopping app?

Answer:

My shopping app allows users to browse products, add items to their cart, and remove items from their cart. It uses React for the front-end, Redux for state management, and `react-hot-toast` for notifications. Users can navigate between the home page and the cart page, where they can see the products they have added.

Question 2: How do you manage state in your application?

Answer:

I manage state using Redux Toolkit. I created a slice for the cart, which contains actions to add and remove items. The state is centralized in the Redux store, and components access and update this state using `useSelector` and `useDispatch` from `react-redux`.

Question 3: How does your app handle adding and removing items from the cart?

Answer:

Each product has buttons to add or remove the item from the cart. When a user clicks "Add to Cart," the `add` action is dispatched, adding the product to the cart state in Redux. Similarly, when "Remove Item" is clicked, the `remove` action is dispatched, removing the product based on its ID. Toast notifications inform the user of these actions.

Question 4: Can you explain how routing is handled in your app?

Answer:

Routing is handled using `react-router-dom`. In `App.js`, I define routes for the home page and the cart page using `Route` components within a `Routes` component. The `Navbar` contains `NavLink` components to navigate between these pages.

Question 5: How do you display loading indicators while fetching data?

Answer:

I use a `Spinner` component to show a loading indicator while data is being fetched. The `Home` component has a `loading` state variable that is set to `true` when the fetch request starts and set to `false` when the data is received. The `Spinner` is conditionally rendered based on the `loading` state.

Question 6: How do you ensure that the cart badge updates correctly in the navbar?

Answer:

The `Navbar` component uses the `useSelector` hook to access the `cart` state from the Redux store. It displays the number of items in the cart using `cart.length`. This ensures that the badge updates automatically whenever items are added or removed.

Question 7: Can you describe how the product data is fetched and displayed on the home page?

Answer:

In the `Home` component, I use the `useEffect` hook to call an asynchronous function, `fetchProductData`, which fetches product data from an API. The data is stored in the `posts` state variable. If the fetch is successful, the product data is mapped over and each product is rendered using the `Product` component. If the fetch fails, an error message is logged, and the `posts` state is set to an empty array.

Question 8: How do you handle errors during the data fetch?

Answer:

In the `fetchProductData` function, I use a try-catch block. If the fetch request fails, the catch block logs an error message to the console and sets the `posts` state to an empty array. This ensures that the UI can handle the failure gracefully by showing a "No Data Found" message instead of crashing.

Question 9: Can you explain the purpose and usage of `react-hot-toast` in your app?

Answer:

`react-hot-toast` is used for displaying toast notifications. In the `Product` and `CartItem` components, I call `toast.success` to show a success message when an item is added to the cart and `toast.error` to show an error message when an item is removed. This provides immediate feedback to users about their actions.

Question 10: How do you style your components?

Answer:

I use Tailwind CSS for styling. Tailwind provides utility classes that I apply directly to my JSX elements. This approach allows for rapid and responsive design adjustments without writing separate CSS files.

---

These questions and answers cover key aspects of your project, from functionality and state management to error handling and user feedback. This should give a comprehensive overview of your project during an interview.